УДК 681.1

О.В. Токаренко, студент гр. ПА-91мп, к.т.н., доц. Богомазов С.А.
КПІ ім. Ігоря Сікорського

СИСТЕМА ЗБОРУ ЕКСПЕРИМЕНТАЛЬНИХ ДАНИХ НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Анотація. У статті наведено аналіз особливостей використання мікросервісної архітектури в програмному забезпеченні мережевих систем збору даних. Визначені основні переваги використання даної архітектури та перспективи її застосування. Систему реалізовано на основі фреймворку Spring WebFlux та нереляційної бази даних MongoDB. Проведений аналіз буде корисним для оптимізації програмного забезпечення систем збору та обробки експериментальних даних.

Ключові слова: мікросервіс, Java, фреймворк Spring, збір та обробка даних, база даних.

ПОСТАНОВКА ПРОБЛЕМИ

В сучасних системах Інтернету речей важливу роль відіграє здатність серверного програмного забезпечення до масштабування, спрощення процесу його розробки і модифікації.

Програмне забезпечення на основі монолітної архітектури будується як єдине ціле, в якому інтерфейс користувача і реалізація доступу до даних об'єднані в одну програму на одній платформі. Труднощі при використанні монолітної архітектури виникають при масштабуванні додатків. Кожен раз, коли розробляються, тестуються та впроваджуються нові функціональні можливості необхідно змінювати весь моноліт, тому що присутня велика зв’язаність між модулями системи.

Мікросервісна архітектура розподіляє додаток на більш дрібні, повністю незалежні компоненти, що забезпечує їм більшу гнучкість і масштабованість. Цей тип архітектури передбачає велику кількість невеликих сервісів, кожен з яких виконує свої власні функції і може бути незалежно розгорнутий. Такі сервіси виконуються в окремих процесах та комунікують між собою через веб-запити або віддалені виклики процедур. При цьому виникає задача загальної організації системи, так як подальша розробка та можливість внесення змін будуть залежати саме від цього. Було проведено аналіз особливостей реалізації такого типу систем та розроблено демонстраційну систему збору експериментальних даних на базі мікросервісної архітектури на основі сучасних Java фреймворків та нереляційної бази даних.

ОСНОВНІ ПОЛОЖЕННЯ

На базі мікросервісного архітектурного стилю [1] розроблена демонстраційна система збору експериментальних даних. Вона складається з наступних компонентів: сервіси, API Gateway, Message broker, MongoDB. На рис.1 наведено структурну схему мікросервісної системи збору даних.

API Gateway. Це служба, що надає єдину точку входу для певних груп мікросервісів. Служба реалізована на основі за допомогою серверу-шлюзу Zuul. Це маршрутизатор і серверний балансувальник навантаження від Netflix, що працює в розробленій системі сумісно з сервером регістрації мікросервісів Eureka.

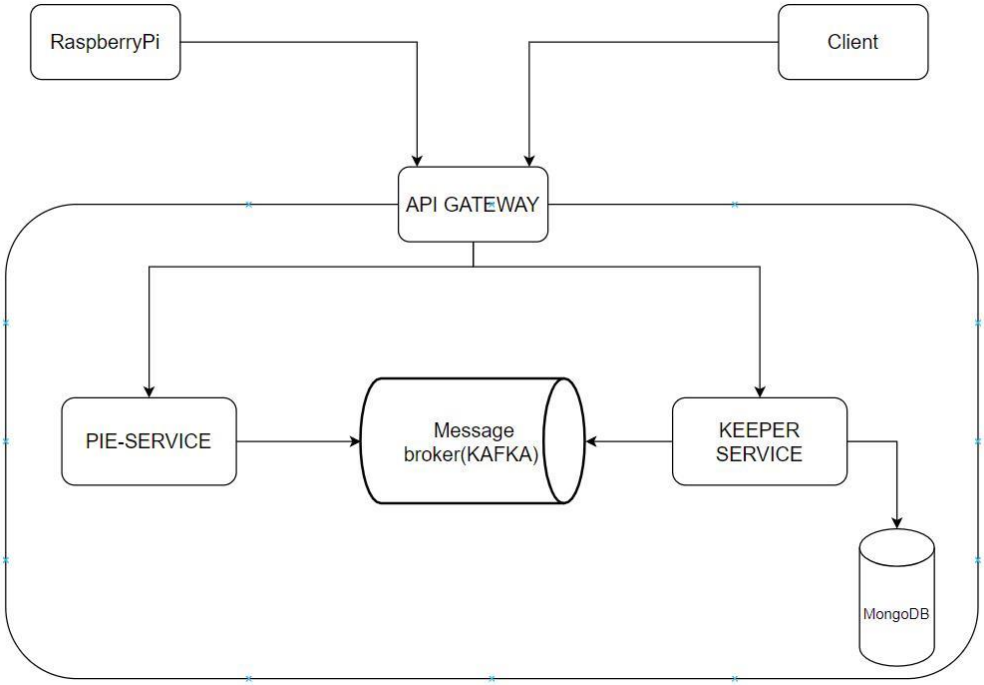

Рисунок 1. Структурна схема системи збору даних на базі мікросервісної архітектури

Message broker (Kafka). За допомогою брокера повідомлень Kafka в системі реалізовано архітектурний шаблон Message broker**.** Kafka – це високопродуктивна розподілена система обміну повідомленнями [2]. Черга повідомлень дозволяє позбутись зв'язаності між мікросервісами. При її використанні немає необхідності знати про особливості інших компонентів системи, все що потрібно – це передати повідомлення до брокера. Таким чином елементи системи можуть бути замінені без порушення загального функціонування системи з мінімальними зусиллями з боку розробників.

MongoDB. Важливою частиною систем збору та обробки експериментальних даних є збереження отриманої інформації в базі даних. Використано MongoDB – документоорієнтовану систему управління базами даних, яка не потребує опису схеми таблиць. Вважається одним з класичних прикладів NoSQL-систем, використовує JSON-подібні документи і схему бази даних. Однією з переваг такого типу баз даних є масштабованість, ефективне збереження великих об'ємів даних, швидкість виконання операцій та вбудована підтримка асинхронного виконання запитів [3].

RaspberryPi. Для реалізації вузлів збору даних було використано одноплатний комп'ютер RaspberryPi та датчики температури та вологості DHT22. Основною завданням програмного забезпечення даного компонента є передача експериментальних даних через API шлюз до мікросервісу Pie-Service.

Для реалізації системи було обрано платформу WebFlux Spring. Платформа WebFlux є альтернативою для версії Spring MVC і реалізує реактивний підхід для створення веб-сервісів [4]. Spring WebFlux реалізує асинхронний і неблокуючий веб-стек, який дозволяє обробляти велику кількість одночасних з'єднань (рис.2).

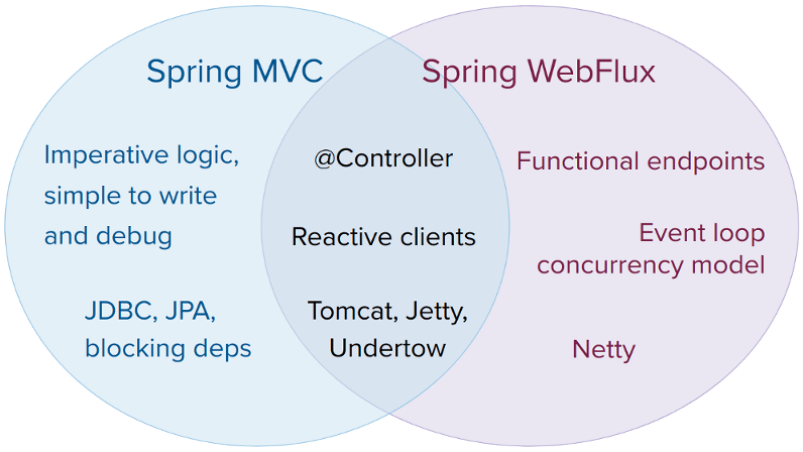

Рисунок 2. Особливості фреймворків Spring MVC та Spring WebFlux

В розробленій системі отримані від сенсорів вимірювальних пристроїв дані надсилаються до шлюзу веб-додатку, який перенаправляє їх до мікросервісу Pie-Service. Pie-Service додає відліки дати та часу та відсилає дані в чергу брокера повідомлень Kafka. Мікросервіс Keeper-Service є підписником на ці повідомлення. При появі нового повідомлення Keeper-Service отримує його, виконує обробку та зберігає до бази даних MongoDB.

Клієнт за запитом отримує дані для вибраного за відповідними координатами геолокації вузла збору даних. При зверненні клієнт відкриває з'єднання з базою даних. Так як MongoDB підтримує з'єднання типу tailable cursor, з'єднання може залишатися відкритим. Це дозволяє отримувати експериментальні дані в режимі, наближеному до реального часу.

ВИСНОВОК

Таким чином, була реалізована демонстраційна вимірювальна система збору експериментальних даних на базі мікросервісної архітектури. Даний підхід дав можливість створити систему з низькою зв'язаністю елементів та широкими можливостями до масштабування та модифікації. Було організовано передачу вимірювальних даних до шлюзу реалізованої веб-системи та їх подальшу обробку, збереження та представлення. В результаті використання реактивного підходу зросли відмовостійкість системи та можливість витримувати великі навантаження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Software architecture. [Online]. Available: https://en.wikipedia.org/wiki/ Software_architecture. Accessed on: November 1, 2020.
[2] Message Broker. [Online]. Available:https://en.wikipedia.org/ wiki/ Message_broker. Accessed on: November 1, 2020.
[3] MongoDB. [Online]. Available:https://ru.wikipedia.org/wiki/MongoDB. Accessed on: November 1, 2020.
[4] Spring WebFlux. [Online]. Available:https://docs.spring.io/spring-framework /docs/current/reference/html/web-reactive.html. Accessed on: November 1, 2020.

Наук. керівник – к.т.н., доц. Богомазов С.А.